Trade-offs

Funzionalità vs Tempo

Non saranno implementate le funzionalità secondarie per presentare il prima possibile il progetto, come ad esempio registrazione, modifica informazioni personali.

Per questo motivo porremo maggiore attenzione alle funzionalità chiave di Sine Charta.

Leggibilità del codice vs Velocità di sviluppo

Il codice dovrà essere comprensibile per semplificare manutenzione e modifiche future. Utilizzeremo quanto più possibile dei commenti per indicare lo scopo delle classi e metodi, rallentando lo sviluppo del sistema.

Scalabilità vs Manutenibilità

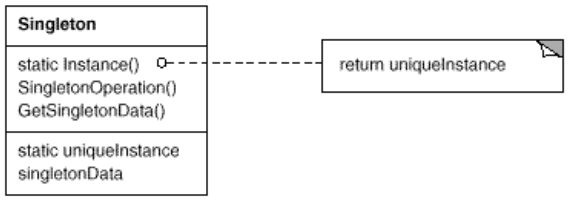
Sine Requie è un gioco in continua evoluzione. Il suo regolamento, infatti, è soggetto a piccoli cambiamenti nel corso degli anni. Sine Charta deve essere progettato in maniera tale da poter tenere il passo con lo sviluppo del gioco reale.

Usabilità vs Sicurezza

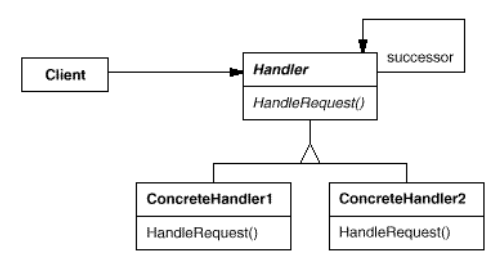
Sine Charta non è e non vuole essere un software per memorizzare dati sensibili. Pertanto ci occuperemo soprattutto di migliorare l'esperienza di gioco dell'utente finale attraverso un sistema usabile, risspetto all'implementazione di sistemi di sicurezza dei dati ad hoc.

Design pattern.

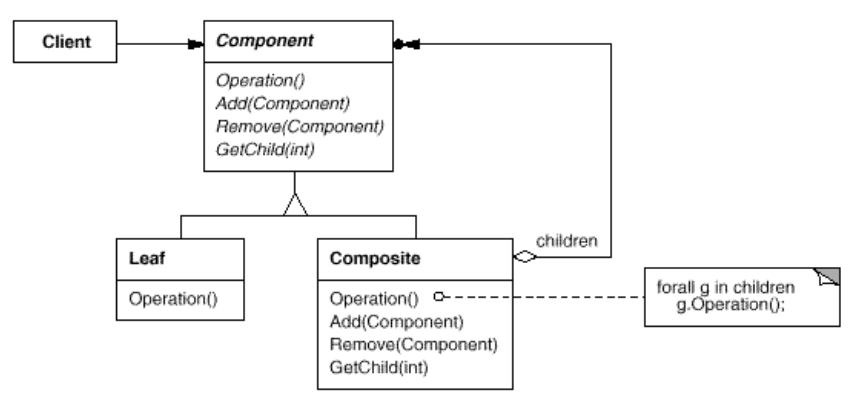
Sine Charta utilizza il Singleton pattern per consentire di istanziare un Manager una e una sola volta.



SineCharta utilizza il Chain of Responsibility pattern durante la creazione del personaggio. Riducendo l'accoppiamento tra le classi che si occupano dell'oggetto Personaggio, sarà possibile facilitare le operazioni di manutenzione e scrittura del codice.



Sine Charta utilizza il Composite Pattern nella rappresentazione dei tarocchi e delle carte da poker. Di fatto, i due mazzi di carte hanno caratteristiche differenti, ma vengono utilizzate allo stesso modo. Con il Composite Pattern potremo separare le due astrazioni delle carte, mantenendo un comportamento di default unico.



|  |  |
| --- | --- |
| **Classe:** | **Descrizione:** |
| utenteRegistratoBean | È l'utente registrato al sistema |
| utenteGiocatoreBean | È l'utente registrato che opera come Giocatore |
| utenteModeratoreBean | È l'utente registrato che agisce come Moderatore |
| MazzoBean | È un insieme di carte dello stesso tipo |
| CartaPokerBean | È una singola carta da gioco che compone il mazzo da poker |
| TaroccoBean | È un singolo tarocco che compone il mazzo dei tarocchi |
| PersonaggioBean | È l'alter-ego del giocatore |
| OggettoBean | È parte dell'equipaggiamento del personaggio |
| ArmaBean | È un particolare oggetto del personaggio |
| StoriaBean | È il contesto immaginario nel quale agiscono i personaggi |
| SessioneBean | È l'insieme di incontri ed eventi descritti dal moderatore |
| KeywordBean | È un evento, una persona o simili che ricorre in una storia |
| NemicoBean | boh |
| AmministratoreBean | È un amministratore del sito web |

Class Interfaces

Package: Gestione utente

Nome: registraUtente(username, password, email, nome, cognome)

@pre !getUtenteById(username)

@pre !getUtenteByMail(email)

@post getUtenteById(username)

@inv getUtenteById(username)!=null || getUtenteByMail(email)!=null

Nome: login(username, password)

@pre getUtenteByLogin(username, password)!=null

@post utente= getUtenteById(username)

@inv !getUtenteByLogin(username, password)

Nome: logout()

@pre utente

@post !utente

@inv !utente

Nome: recuperaCredenziali(email)

@pre uTemp= getUtenteByMail(email)

@post uTemp.user

@post uTemp.password

@inv !getUtenteByMail(email)

Nome: modificaInfo(param, newParam)

@pre utente.param

@post utente.param= newParam

@inv newParam==param

Package: Gestione Storia

Nome: Creazione Storia (Nome, Descrizione, Ambientazione)

@pre !getStoria(Nome)

@post getStoria(Nome)

@inv getStoriabyNome(Nome) != null

@inv getStoriabyDescrizione(Descrizione) != null

Nome: Crezione Keyword(Chiave, Descrizione)

@pre !getKeyword(Chiave)

@post getKeyword(Chiave)

@inv getKeywordbyId(Chiave) != null

Nome: Invitare utentiGiocatori alla Storia( Storia, utenteGiocatore.username)

@pre getUtenteById(username)!=null

@pre getStoriabyNome(Storia.nome) != null

@post isInvioSpedito() [da controllare]

Nome: Accettare un invito(invito)

@pre getInvito() != null

@post invitoAccettato() == true

@inv

Nome: Creare un PG(Generalità, Tarocco\_dominante, Caratteristiche, Abilità, Risoluzione, Ferite)

@pre !getPG(Generalità)

@post getPG(Generalità)

@inv getPGbyId(Generalità) != null

Nome: Partecipare ad una Storia(Storia, utenteGiocatore)

@pre getStoria()

@post AddUser(utenteGiocatore) == true

@inv [da finire]